A Bootstrap Approach to Automatically Generating Lexical Transfer Rules

Davide Turcato Paul McFetridge Fred Popowich and Janine Toole
Natural Language Laboratory, School of Computing Science, Simon Fraser University
8888 University Drive, Burnaby, British Columbia, V5A 1S6, Canada
and
Gavagai Technology
P.O. 374, 3495 Cambie Street, Vancouver, British Columbia, V5Z 4R3, Canada
{turk,mcfet,popowich,toole}@cs.sfu.ca

Abstract

We describe a method for automatically generating Lexical Transfer Rules (LTRs) from word equivalences using transfer rule templates. Templates are skeletal LTRs, unspecified for words. New LTRs are created by instantiating a template with words, provided that the words belong to the appropriate lexical categories required by the template. We define two methods for creating an inventory of templates and using them to generate new LTRs. A simpler method consists of extracting a finite set of templates from a sample of hand coded LTRs and directly using them in the generation process. A further method consists of abstracting over the initial finite set of templates to define higher level templates, where bilingual equivalences are defined in terms of correspondences involving phrasal categories. Phrasal templates are then mapped onto sets of lexical templates with the aid of grammars. In this way an infinite set of lexical templates is recursively defined. New LTRs are created by parsing input words, matching a template at the phrasal level and using the corresponding lexical categories to instantiate the lexical template. The definition of an infinite set of templates enables the automatic creation of LTRs for multi-word, non-compositional word equivalences of any cardinality.

1 Introduction

It is well-known that Machine Translation (henceforth MT) systems need information about the different ways in which words can be translated, depending on their syntactic and semantic context. Such lexical transfer rules (henceforth LTRs) are notoriously time-consuming for humans to construct. Our task is to automatically generate LTRs.

An LTR can be seen as a word equivalence plus an associated transfer pattern. By *word equivalence* we mean a translation pair simply stated in terms of words, in a dictionary-like fashion. A transfer pattern specifies how transfer is to be performed for each of the morphological variants of the words in the equivalence and for different syntactic contexts. For instance, given an English-Spanish word equivalence

get lucky ↔ *tener suerte*

the associate transfer pattern would have to account for all the following equivalences:

I *get lucky*	*Tengo suerte*
I *will get lucky*	*Tendré suerte*
I *would have got lucky*	*Habría tenido suerte*
Getting lucky	*Teniendo suerte*
I start *getting lucky*	Empiezo a *tener suerte*
I start *getting very lucky*	Empiezo a *tener mucha suerte*

where the Spanish sentences can be glossed as follows:

a. *Tengo* *suerte*
I have good luck

b. *Tendré* *suerte*
I will have good luck

c. *Habría* *tenido* *suerte*
I would have had good luck

d. *Teniendo* *suerte*
Having good luck

e. Empiezo a *tener* *suerte*
I start to have good luck

f. Empiezo a *tener* *mucha* *suerte*
I start to have much good luck

The last example in the list of translation pairs shows that a transfer pattern also has to account for modifiers. In the example, *very* is translated by the adjective *mucha*, whereas in a sentence like *I start getting very lazy* it would be translated by the adverb *muy* (*Empiezo a volverme muy perezoso*).

Thus, a bilingual lexicon of such transfer rules is a different object from a collection of word equivalences, which is the definition of a bilingual lexicon most often found in the literature about automatic creation of bilingual lexicons ([3], [4], to cite recent examples). As a matter of fact, those techniques and the one described here are disjoint and complementary, as the output of those tools can be used as input to LTR development. Given a collection of word equivalences, we focus on how transfer patterns can be associated with them to create complete LTRs.

This task has rarely been tackled before. Several techniques have been proposed for the automatic acquisition of word equivalences (see references above), but very few for the automatic acquisition of full LTRs (e.g. [1]), despite the high cost of their manual development. Bilingual coding is often a bottleneck in MT system development. Unlike other linguistic resources, like grammars, lexicons have an open-ended linear growth and their quality can be directly related to their size. For this reason, there is often a mismatch between the development time frames of bilingual lexical resources and other modules.

2 Basic ideas

2.1 Template based generation

We use a bootstrap approach to transfer rule creation. An initial hand coded bilingual lexicon is used as a basis for defining a set of transfer rule *templates*, i.e. skeletal rules unspecified for words. Subsequently, appropriate transfer rule templates are associated to new word equivalences, on the basis of the morphosyntactic features of those words, to construct complete LTRs. The approach described here shares this underlying template-based bootstrap philosophy with the approach described by [6], but differs from it in three key respects: the resources it uses, the way templates are created and the way LTRs are created from word equivalences and templates. LTR templates are also akin to *tlinks*, as described in [1] and [2].These works describe how to use tlinks for the semi-automatic generation of single-word equivalences. However, they do not deal with the creation of an inventory of tlink types or the generation from multi-word equivalences.

For the sake of exposition, we use here a simplified version of LTRs and templates, showing only words (for LTRs), syntactic categories and indices, the latter represented by tuples of subscript lowercase letters. A schematic LTR and template are shown in (1) and (2), respectively. For a description of the MT system and the full LTR formalism see [5] and [7].

Templates	LTRs	Coverage
1	5683	33.9 %
2	8726	52.1 %
3	10710	63.9 %
4	12336	73.6 %
5	13609	81.2 %
50	15473	92.3 %
500	16338	97.5 %
922	16760	100.0 %

Table 1: Incremental template coverage

(1) W^{s1}:$\texttt{Cat}^{s1}_{Indices^{s1}}$ & ...& W^{sm}:$\texttt{Cat}^{sm}_{Indices^{sm}}$
$\leftrightarrow$
W^{t1}:$\texttt{Cat}^{t1}_{Indices^{t1}}$ & ...& W^{tn}:$\texttt{Cat}^{tn}_{Indices^{tn}}$

(2) $\texttt{Cat}^{s1}_{Indices^{s1}}$ & ...& $\texttt{Cat}^{sm}_{Indices^{sm}}$
$\leftrightarrow$
$\texttt{Cat}^{t1}_{Indices^{t1}}$ & ...& $\texttt{Cat}^{tn}_{Indices^{tn}}$

Given a word equivalence expressing a translation pair, our goal is to create a transfer rule directly usable by an MT system. In other words, the goal is to associate a transfer pattern, as informally described above, to a word equivalence. We describe two approaches, the latter of which is an extension of the former.

2.2 The enumerative approach

The goal of creating templates and using them in generating LTRs can be accomplished through the following steps:

1. Create transfer rule templates:
   (a) Define a set of LTR templates. Each template represents a transfer pattern. The template definition task is carried out by extracting LTR templates from an initial hand-coded bilingual lexicon. This can be easily done by simply removing words from LTRs, normalizing variables by renaming them in some canonical way (so as to avoid two instances of the same templates to only differ by variable names), then ranking templates by frequency, if the application of some cutoff is in order. Table 1 shows the incremental coverage of the set of templates we extracted from our initial hand-coded English-Spanish bilingual lexicon. We refer to the template creation process described here as the *enumerative* approach to building templates.
   (b) Associate a set of constraints to each template. Typically, these are morphosyntactic constraints on the words to be matched against the template. Basically, such constraints ensure that an input word belongs

to the same lexical category of the template item it has to match. The same goal could also be achieved by directly unifying a lexical category associated to a word with the corresponding template item, instead of having separate constraints.

2. Create transfer rules:

   (a) Given a word equivalence, create an LTR if the lexical descriptions of the words in the translation pair satisfy all the constraints associated with a template (or unify with their corresponding items in the template). In that case, an LTR is created by simply instantiating the successful template with the words in the word equivalence.

An enumerative approach to template creation guarantees an adequate coverage for creating most of the LTRs needed in an MT system, as discussed in [6]. A simple LTR automatic generation procedure can be implemented by selecting the most significant templates in the database. This can be done, as hinted above, by counting the occurrences of each template in the LTR corpus and choosing those that rank best. The top ranking templates are then used to directly map input word equivalences onto LTRs. This approach was implemented and used, with good results (e.g. in an early test run on a 1544 entry word-list downloaded from the World Wide Web, LTRs were created for 79% of the input word equivalences. Further results are discussed in section 4.

2.3 The generative approach

The idea of adding recursion to the template definition procedure, thus replacing a finite set of templates with an infinite one, was brought about by work on phrasal verbs. Phrasal verbs exhibit a larger variability than other collocations. One of the problems is that their translations are often paraphrases, because a target language might lack a direct equivalent to a source phrasal verb. Table 2 illustrates this point (e.g. *sit through something* ↔ *permanecer hasta la fin de algo*).

A finite set of templates could still be used for phrasal verbs, but this would require a much larger initial LTR corpus than is necessary for other collocations, in order to preserve a high automatic generation rate. An alternative solution is based on introducing a further level of abstraction, by defining higher level, underspecified templates which state bilingual equivalences in terms of phrasal categories instead of lexical categories. Then, a simple grammar is used to map such phrasal categories onto sets of lexical categories in order to derive completely specified templates.

Despite the template variability in terms of sequences of syntactic categories, a much higher regularity can be found by defining templates in terms of constituency. For instance, all the lexical equivalences listed for *sit* in Table 2 can be reduced to two basic patterns in terms of phrasal categories:[1]

(3) `VP` ↔ `VP`

(4) `VP/NP` ↔ `VP/NP`

where `VP/NP` represents a verbal phrase with a noun phrase gap. A further generalization is that a `VP` on either side of a template tends to be equivalent to a phrase of the same type, with the same number and type of gaps. We note incidentally that this further generalization does not hold for all categories, in terms of category identity. For example, an English adjective often corresponds to a Spanish prepositional phrase (e.g. *fashionable* ↔ *de moda*, *stainless* ↔ *sin tacha*).

The abstraction process consists of partitioning lexical templates into classes such that each class is identified by a phrasal template, in which a group of lexical categories is replaced by a phrasal category. All the lexical templates in a class can be obtained by replacing the phrasal category with one of its lexical projections. Such replacement can be carried out on a purely monolingual basis, by using a simple grammar to define constituency. The key requirement on the abstraction process is that the resulting abstract template be invariant with respect to lexical replacement, i.e. the replacement do not involve any other element in the abstract template, beside the replaced phrasal category. This restriction amounts to requiring that a phrasal category be self-contained in terms of variable sharing, i.e. the lexical categories it dominates introduce no new variables to be shared with items external to the phrase itself; or, if such sharing happens, it must be entirely predictable and unambiguous, i.e. the variable sharing lexical category must be marked in such a way that it can be uniquely identified.

Once a phrasal template has been defined, and a grammar is available for mapping phrasal categories onto lexical categories, new, previously unseen lexical templates can be derived that were not present in the initial class which the phrasal template was abstracted over. Depending on the recursivity of the grammars in use, an infinite set of lexical templates can be defined via a finite set of phrasal template. Therefore, we refer to this process as to a *generative* approach to building templates. For example, the templates underlying the bilingual entries in (5) allow one to infer the template underlying the bilingual entry for (6).

(5) *Buddha* ↔ *buda*
Wonderland ↔ *país de las maravillas*

(6) *Halloween* ↔ *víspera del Día de los Santos*

Namely, from the lexical templates in (7), the phrasal template in (8) can be inferred. In turn, the new

[1] We adopt the convention of using lowercase labels for lexical categories and uppercase labels for phrasal categories.

Phrasal verb equivalences:

sit back	↔ *ponerse cómodo*	`iv adv`	↔ `rv adj`
sit down	↔ *sentarse*	`iv adv`	↔ `rv`
sit for sth	↔ *posar para* algo	`iv p`	↔ `iv p`
sit in for sb	↔ *sustituir* algn	`iv adv p`	↔ `tv`
sit in on sth	↔ *participar como observador en* algo	`iv adv p`	↔ `iv p n p`
sit through sth	↔ *permanecer hasta la fin de* algo	`iv p`	↔ `iv p det n p`
sit tight	↔ *esperar*	`iv adj`	↔ `iv`
sit up with sb	↔ *velar* algn	`iv adv p`	↔ `tv`
sit up	↔ *incorporarse*	`iv adv`	↔ `rv`
sit sb *down*	↔ *sentar* algn	`tv adv`	↔ `tv`
sit sb *up*	↔ *incorporar* algn	`tv adv`	↔ `tv`
sit sth *out*	↔ *no participar en* algo	`tv adv`	↔ `neg iv p`

Glosses for Spanish:

a. *ponerse* *cómodo*
put oneself comfortable

b. *sentarse*
sit oneself

c. *posar* *para* algo
pose for sth

d. *sustituir* algn
replace sb

e. *participar* *como* *observador* *en* algo
take part as observer in sth

f. *permanecer* *hasta* *la* *fin* *de* algo
remain until the end of sth

g. *esperar*
wait

h. *velar* algn
watch over sb

i. *incorporarse*
raise oneself

j. *sentar* algn
sit sb

k. *incorporar* algn
raise sb

l. *no* *participar* *en* algo
do not take part in sth

where:
En. sth = Sp. algo = something
En. sb = Sp. algn = somebody

Table 2: Some phrasal verb equivalences for the verb *sit*, and associated glosses for Spanish.

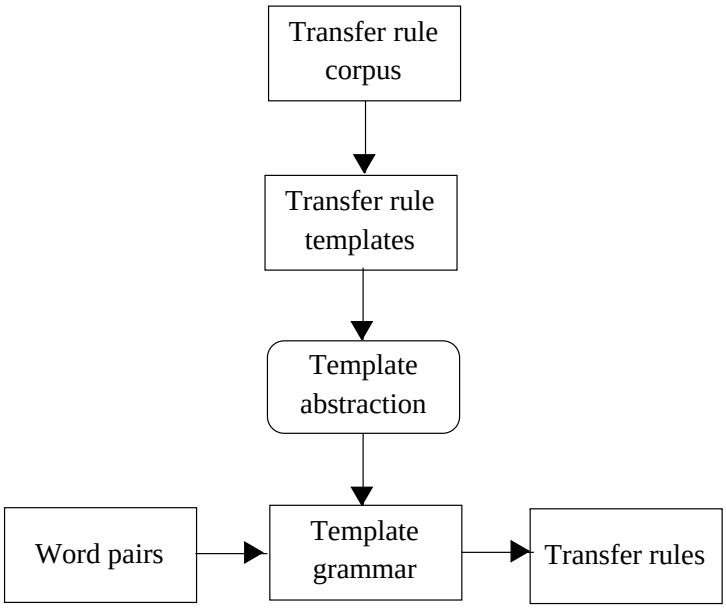

Figure 1: System architecture.

lexical template in (9) can be derived from (8), provided that the relevant grammar licenses the projection of a phrasal category NBAR onto an appropriate sequence of lexical categories.

(7) $\mathtt{n}_a \leftrightarrow \mathtt{n}_a$
$\mathtt{n}_a \leftrightarrow \mathtt{n}_a \ \mathtt{\&} \ \mathtt{p}_{a,b} \ \mathtt{\&} \ \mathtt{d}_b \ \mathtt{\&} \ \mathtt{n}_b$

(8) $\mathtt{n}_a \leftrightarrow \mathtt{NBAR}_a$

(9) $\mathtt{n}_a \leftrightarrow \mathtt{n}_a \ \mathtt{\&} \ \mathtt{p}_{a,b} \ \mathtt{\&} \ \mathtt{d}_b \ \mathtt{\&} \ \mathtt{n}_b \ \mathtt{\&} \ \mathtt{p}_{b,c} \ \mathtt{\&} \ \mathtt{d}_c \ \mathtt{\&} \ \mathtt{n}_c$

The overall architecture of the generative approach is shown in Figure 1. The idea of the whole process is to exploit monolingual regularities to account for a non-compositional bilingual equivalence. In a non-compositional equivalence, direct correspondences between lexical items on either side cannot be established. The two sides are only equivalent as wholes. However, a non-compositional equivalence can be accounted for by decomposing it into a phrasal bilingual equivalence and monolingual mappings from phrasal to lexical categories.

Implementing a template grammar allows one to obtain an adequate template coverage while requiring only a small initial LTR corpus. In our system the template abstraction task was performed manually. Although the implementation of an automatic template abstraction procedure could be foreseen, the high reliability required of templates demands that a strict human control still be placed at some point of the template abstraction and grammar definition phases. It is also worth pointing that, by our experience, the grammar development task is not very labour intensive. Such grammars have to perform a very limited task, and deal with a restricted and controlled input. Basically, they only have to account for the constituent structure of very simple and syntactically ordinary phrases. In our case, the grammar development time was usually measured in hours.

3 Implementation

In this section the generation of an LTR from a phrasal template is described. We illustrate the procedure with the aid of a worked example. We show how the LTR in (11) is generated from the word equivalence in (10).

(10) *sit in on* sth ↔
participar como observador en algo

(11) $\mathit{sit}\mathtt{:iv}_{a,b,c} \ \mathtt{\&} \ \mathit{in}\mathtt{:adv}_c \ \mathtt{\&} \ \mathit{on}\mathtt{:p}_{a,d} \leftrightarrow$
$\mathit{participar}\mathtt{:iv}_{a,b,e} \ \mathtt{\&} \ \mathit{como}\mathtt{:p}_{e,f} \ \mathtt{\&}$
$\mathit{observador}\mathtt{:n}_f \ \mathtt{\&} \ \mathit{en}\mathtt{:p}_{a,d}$

Pragmatic reasons induced us to take a hybrid approach to the template grammar construction task, combining the enumerative and generative approaches to build templates. It turned out that templates show a larger variability on the Spanish side than on the English side. Table 2 is a good example of this point. This fact might be related to a larger use of concise and productive phrasal verb patterns in English, or perhaps to the fact that our word equivalence coding was driven by English collocations, thus involving the presence of paraphrases on the Spanish side only. In any case it appeared that the enumerative approach was sufficient to adequately cover the English

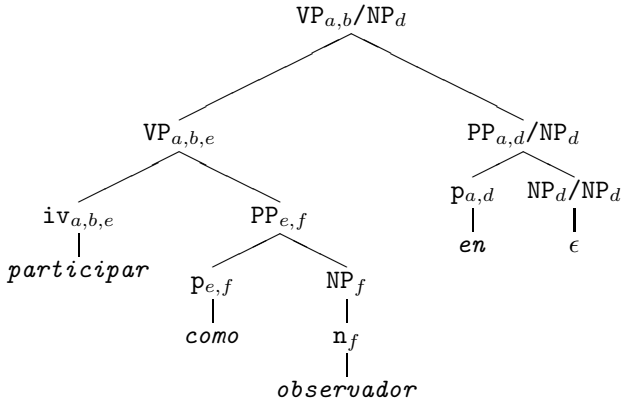

Figure 2: Template right hand side generation tree.

side without resulting in a proliferation of templates. Hence, we constructed a set of templates in which only the Spanish side contained a phrasal category, while the English side was fully specified. For example:

(12) $\mathtt{iv}_{a,b,c}$ & $\mathtt{adv}_c$ & $\mathtt{p}_{a,d} \leftrightarrow \mathtt{VP}_{a,b}/\mathtt{NP}_d$

For each such partially specified template, the Spanish side has to be generated.

The selection of a phrasal template candidate is performed by doing a lexical lookup for the source words and matching their morphosyntactic representations with the corresponding left hand side template items. This can be done directly by unification or by associating a set of constraints to the phrasal template. In our implementation, constraints are expressed by Prolog goals taking morphosyntactic descriptions as arguments. For example, the phrasal template in (12) is selected as a candidate for our input translation pair in (10), as the lexical descriptions of *sit*, *in* and *on* match the categories iv, adv and p, respectively. A sequence of words can match several phrasal templates, e.g. in our specific example the following candidate is also selected, since *in* can also be a noun (e.g. *the ins and outs of a problem*):

(13) $\mathtt{tv}_{a,b,c}$ & $\mathtt{n}_c$ & $\mathtt{p}_{c,d} \leftrightarrow \mathtt{VP}_{a,b}/\mathtt{NP}_d$

Given a candidate phrasal template, the core of the generation procedure is a call to a target language grammar, Spanish in this case, which parses the target input words using the given phrasal category (with its associated indices) as its initial symbol.

A phrasal template may disjunctively specify several phrasal categories as initial symbols (or, equivalently, different phrasal templates may share the same English side, while specifying different initial symbols). For instance, English adjectives may be equivalent to either a Spanish adjectival phrase or prepositional phrase, as already mentioned.

Figure 2 shows the result of parsing the Spanish input words of our example, using the initial symbol in the phrasal template in (12). Each node shows the assigned syntactic category, along with the indices, either specified in the initial symbol or instantiated during parsing.

The pre-terminal categories in the parse tree, along with the input words, are used to build the following LTR right hand side for the final LTR:

(14) *participar*:$\mathtt{iv}_{a,b,e}$ & *como*:$\mathtt{p}_{e,f}$ & *observador*:$\mathtt{n}_f$ & *en*:$\mathtt{p}_{a,d}$

Finally, by instantiating the left hand side of the template with the input English words and replacing the right hand side phrasal category in the phrasal template (12) with the right hand side in (14), the LTR in (11) is obtained.

Again, several output LTRs can be generated for the same input words. For example, the following LTRs are also generated for our example:

(15) *sit*:$\mathtt{iv}_{a,b,c}$ & *in*:$\mathtt{adv}_c$ & *on*:$\mathtt{p}_{a,d} \leftrightarrow$ *participar*:$\mathtt{iv}_{a,b}$ & *como*:$\mathtt{p}_{a,e}$ & *observador*:$\mathtt{n}_e$ & *en*:$\mathtt{p}_{a,d}$

(16) *sit*:$\mathtt{iv}_{a,b,c}$ & *in*:$\mathtt{adv}_c$ & *on*:$\mathtt{p}_{a,d} \leftrightarrow$ *participar*:$\mathtt{iv}_{a,b,e}$ & *como*:$\mathtt{p}_{e,f}$ & *observador*:$\mathtt{n}_f$ & *en*:$\mathtt{p}_{f,d}$

In (15) the indices show that *como* is analyzed as a modifier of *participar*, instead of a complement. In (16) the preposition *en* modifies *observador*, instead of *participar*. In our case, since we are interested in translating only from English to Spanish, we would keep only one of the candidate LTRs shown above, as they all share the same English side. However, given that the LTRs at hand are bidirectional, if one were interested in translating from Spanish to English, one might want to keep all the entries, in order to cover different syntactic analyses.

The last step is the validation of LTRs by lexicographers, which exclusively consists of removing unwanted entries. Lexicographers use their linguistic intuition and knowledge of the syntactic representations

File	In	Out	Val	InOut	InVal	%
Enumerative approach:						
ADJ	542	546	469	468	468	86.3 %
Phrasal verbs - batch 1	2340	2395	1416	1647	1414	60.4 %
Generative approach:						
Phrasal verbs - batch 2	549	1152	486	512	469	85.4 %
Phrasal verbs - batch 3	478	782	404	418	393	82.2 %
V + (ADJ or N)	345	617	300	302	292	84.6 %
ADJ + N	1144	1331	914	914	903	78.9 %
IV + ADJ	199	346	187	187	187	94.0 %

Table 3: Overview of system performance.

used in LTRs, in order to make a choice among several competing LTRs for a given translation pair, or to to check the correctness of the analysis underlying an LTR. Information that they are typically expected to check is the way coindexing is performed (e.g. for prepositions, in order to check that they be attached to the correct item) and the syntactic categories assigned to words, most crucially when unknown words are involved. In our implementation lexicographers are helped in the task by messages that the system associates to candidate LTRs, in order to signal, for instance, the presence of lexically unknown words or lexical ambiguities which are potential sources of errors (for instance, verbs which are both transitive and intransitive). The files output by the system tend to be self-contained in terms of information needed by lexicographers to perform their task, and usually lexicographers do not need access to any extra linguistic resources. We finally note that the validation step becomes particularly crucial if the input translation pairs are automatically acquired from resources like bilingual corpora, in order to filter out LTRs created from noisy bilingual equivalences.

4 Performance

Results of this approach are shown in Table 3. Columns are to be interpreted as follows:

File : The type of words or collocations in the processed file.

In : The number of input word equivalences.

Out : The number of generated LTRs.

Val : The number of LTRs validated by the lexicographers and thus added to the transfer lexicon.

InOut : The number of input word equivalences (**In**) for which some output (**Out**) was provided. This value is usually lower than the value of **Out** because more than one LTR can be created for the same word equivalence. Therefore, more than one element of **Out** can correspond to one element of **InOut**.

InVal : The number of input word equivalences (**In**) for which some LTR was validated (**Val**) by the lexicographers.

% : The success rate, obtained by dividing the value of **InVal** by the value of **In**. By using the value of **InVal** instead of the value **Val** we factor out the extra valid LTRs that can be created for a given input word equivalence, in addition to the first one.

The listed files are sorted according to the chronological order in which they were processed, and divided according to the methodology in use. One of the most common reasons for failure is the presence of unknown words on the English side. When the input contains unknown words, we block generation. In contrast unknown words are accepted on the Spanish side, matching any possible lexical category. This treatment of Spanish unknown words is one of the reasons that explains the higher value in **Out** than in **InOut**. Genuine syntactic ambiguity is the other main reason for such difference in values.

In terms of speed, we ran a test by evaluating the development time of the file named 'Phrasal verbs - batch 1' in Table 3. We had a lexicographer timing three activities: coding translation pairs, revising automatically generated LTRs, manually coding LTRs for the translation pairs that failed in automatic generation. The results are shown in Table 4.

5 Conclusion

The described methodology makes LTR coding considerably faster. According to our test, revising automatically generated LTR files is about 8 times as fast as manually coding LTRs. If the manual coding of word equivalences is also counted in the automatic generation process, the process is still about 3 times as fast as the manual coding of complete LTRs.

Besides speed, this methodology guarantees the syntactic correctness of the output (provided that the templates are syntactically correct, of course). The validation procedure only requires removing unwanted

Activity	N. of items	Items per hour	Time for 100 items
Coding translation pairs	2340	31.25	3h 12m
Revising LTRs	1416 (validated)	50.57	1h 59m
Manually coding LTRs	926	6.25	16h 00m

Table 4: Speed test results.

LTRs, with no further editing intervention. Also, more control over a transfer rule database is provided, as each LTR can be associated with a template. In this way testing, debugging, and maintenance in general are easier and more effective, as these processes can be performed on templates rather than on LTRs.

Although the described methodology can hardly aspire to completeness, compared to manual coding, it can be integrated with a manual coding phase, for the translation pairs that fail to generate automatically. The gain in terms of labour effort is still proportional to the success rate of the generation procedure, when compared to an entirely manual coding. Moreover, there is one sense in which automatic generation is more complete than manual coding, namely in the generation of multiple entries for a translation pair. Lexicographers tend to code one LTR per translation pair, whereas the range of candidates proposed by automatic generation can make them aware of valid alternatives analyses unnoticed by them.

Lexicographic work is easier, as little technical knowledge is required of lexicographers. Full knowledge of the formalism in use for LTRs is only required in the initial phase of LTR corpus constructions. Once a template generation procedure is in place, lexicographers only need a passive knowledge of the formalism, i.e. they must be able to read and understand LTRs, but not to write LTRs.

Input word equivalences are expressed in plain natural language. This makes them amenable to acquisition from corpora or MRDs. If manual coding of word equivalences is necessary, only bilingual speaking competence is required of lexicographers. No further linguistic background or familiarity with any formalism is necessary.

The bootstrap approach makes this methodology more suitable for the scaling up of large scale MT systems, than for rapid development of prototypes. The knowledge acquired in prototype development is used in developing a full-fledged system, which is often the most critical phase for MT systems. The adoption of this methodology is also profitable in porting an MT system to a different language pair or in developing a multilingual MT system: the methodology is applicable to any language pair, and the linguistic knowledge can also be re-used to some extent, depending on the similarity between the languages at hand.

References

[1] Ann Copestake, Bernie Jones, Antonio Sanfilippo, Horacio Rodríguez, Piek Vossen, Simonetta Montemagni, and E. Marinai. Multilingual lexical representation. Technical Report 253, University of Cambridge Computer Laboratory, Cambridge, UK, 1992.

[2] Ann Copestake and Antonio Sanfilippo. Multilingual lexical representation. In *Proceedings of the AAAI Spring Symposium on Building Lexicons for Machine Translation*, Stanford, California, USA, 1993.

[3] Pascale Fung. A statistical view on bilingual lexicon extraction: From parallel corpora to nonparallel corpora. In *Proceedings of the Third Conference of the Association for Machine Translation in the Americas (AMTA-98)*, pages 1–17, Langhorne, Pennsylvania, USA, 1998.

[4] I. Dan Melamed. Empirical methods for MT lexicon development. In *Proceedings of the Third Conference of the Association for Machine Translation in the Americas (AMTA-98)*, pages 18–30, Langhorne, Pennsylvania, USA, 1998.

[5] Fred Popowich, Davide Turcato, Olivier Laurens, Paul McFetridge, J. Devlan Nicholson, Patrick McGivern, Maricela Corzo-Pena, Lisa Pidruchney, and Scott MacDonald. A lexicalist approach to the translation of colloquial text. In *Proceedings of the 7th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 76–86, Santa Fe, New Mexico, USA, 1997.

[6] Davide Turcato. Automatically creating bilingual lexicons for Machine Translation from bilingual text. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL'98)*, pages 1299–1306, Montréal, Québec, Canada, 1998.

[7] Davide Turcato, Olivier Laurens, Paul McFetridge, and Fred Popowich. Inflectional information in transfer for lexicalist MT. In *Proceedings of the International Conference 'Recent Advances in Natural Language Processing' (RANLP-97)*, pages 98–103, Tzigov Chark, Bulgaria, 1997.